| **Method used** | **Dataset size** | **Testing-set predictive performance** | **Time taken for the model to be fit** |
| --- | --- | --- | --- |
| XGBoost in Python via scikit-learn and 5-fold CV | 100 | 0.92 | 0.14 |
| 1000 | 0.9470 | 0.22 |
| 10000 | 0.9646 | 0.45 |
| 100000 | 0.9711 | 6.84 |
| 1000000 | 0.9713 | 29.31 |
| 10000000 | 0.9716 | 303.2369 |
| XGBoost in R – direct use of xgboost() with simple cross-validation | 100 | 0.862 | 0.768 |
| 1000 | 0.929 | 1.573 |
| 10000 | 0.967 | 2.66 |
| 100000 | 0.982 | 8.19 |
| 1000000 | 0.985 | 64.68 |
| 10000000 | 0.987 | 623.66 |
| XGBoost in R – via caret, with 5-fold CV simple cross-validation | 100 | 0.9 | 0.59 |
| 1000 | 0.97 | 2.41 |
| 10000 | 0.969 | 4.23 |
| 100000 | 0.983 | 15.74 |
| 1000000 | 0.987 | 95.66 |
| 10000000 | 0.988 | 786.944 |

I recommend using XGBoost in R via caret with 5-fold cross validation as the most suitable solution for the provided performance metrics. This solution creates the most effective combination of better predictions together with suitable computational resources at all data collection points. The caret implementation delivers the best accuracy scores of 0.988 when using 10 million samples while providing execution times comparable to direct R implementation. The Python scikit-learn provides the fastest execution times but delivers model performance results that are persistently lower compared to all dataset sizes creating major implications for quality prediction results.

The direct xgboost() implementation in R shows similar predictive performance to the caret approach, but with notably longer execution times, particularly at larger scales. The direct xgboost() implementation requires 623.66 seconds to process 10 million samples while taking 303.24 seconds for the Python implementation yet produces slightly better accuracy results (0.987 vs 0.9716). The caret implementation with 5-fold CV achieves the best predictive performance of 0.988 while maintaining execution times that are longer than Python's but still practical for the performance advantages.

Production environments should utilize the R caret implementation because its minor computational overhead is offset by superior predictive model performance. The caret package provides additional functionality such as standardized preprocessing and hyperparameter tuning features and interfaces that span across different modeling techniques because it is more adaptable for complex machine learning operations. The Python scikit-learn implementation continues to serve applications that require high speed performance even when accuracy needs to be slightly compromised.